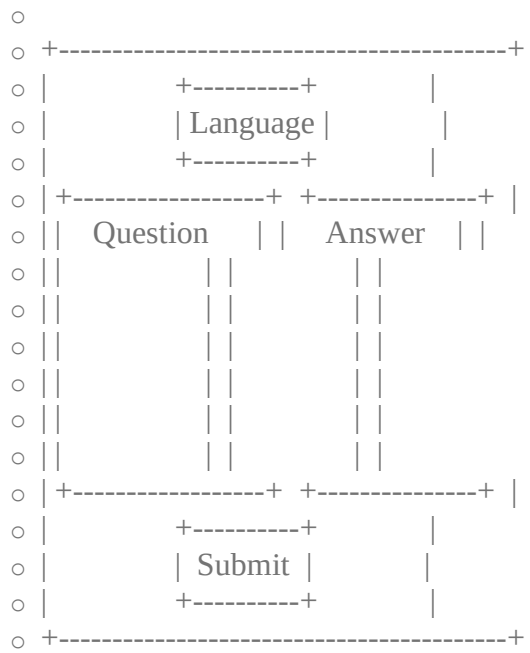
YIN YIN PHYO
20107
Week 4 Homework 1 : Customer Support System: An email to the customer

1. Customer Support System: An email to the customer
    A. Prerequiste
        - If possible, complete Project: Customer Support System: Use ChatGPT to build a web-based system that can answer questions about a website.
    B. Overview
        - If you're a customer service assistant for a large electronics store
            - The website of the store allows the customers to select language.
            - The store's products
                - The products belong to different categories
                - Each product has detailed description
        - The web interface for this project



References

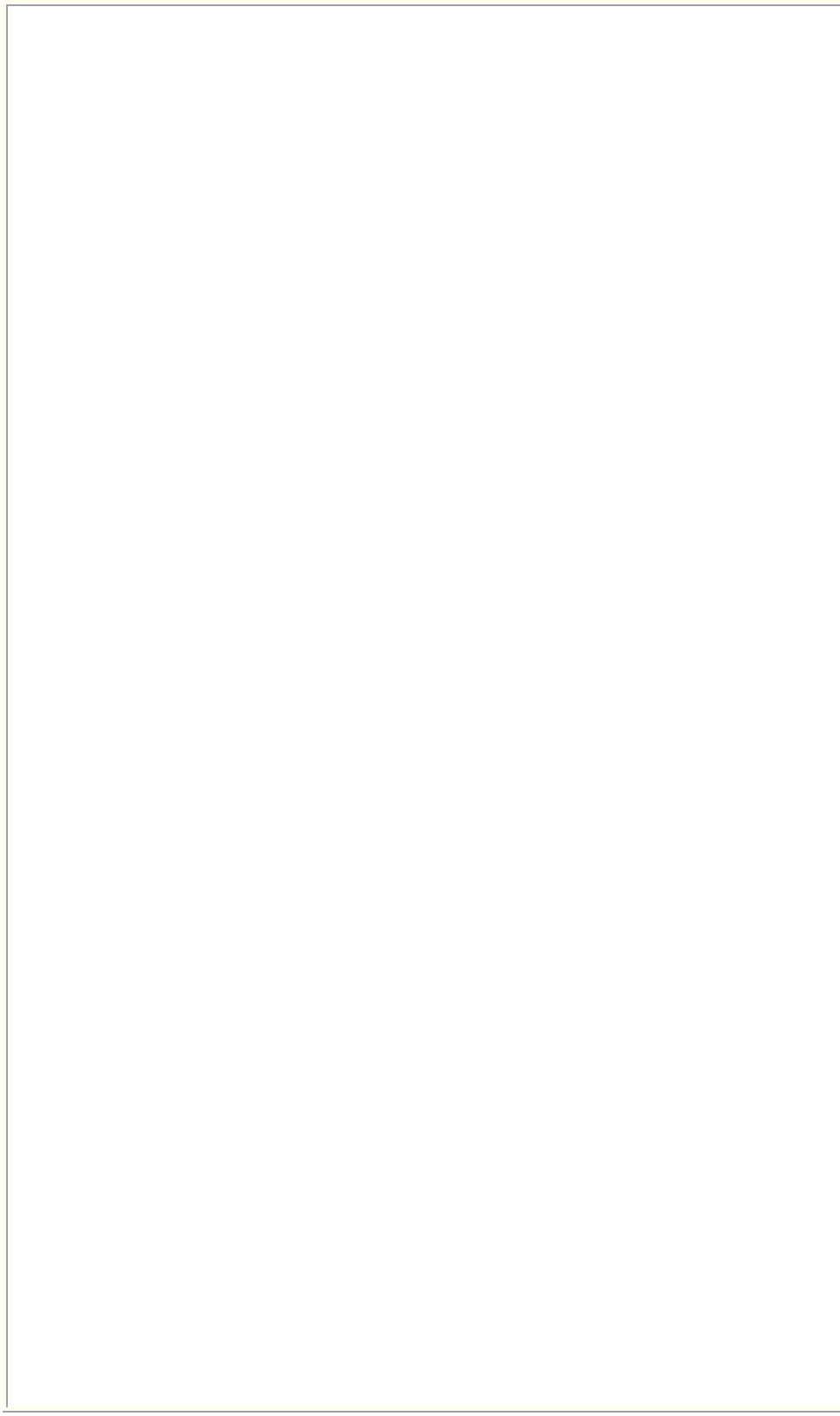
- Spec
    - Question
        - The result of Step 1
    - Answer
        - The result of Step 5
    - Language
        - The language of the Answer.
        - Test cases

ID	Question	Answer
1	English	English
2	English	Non-English
3	Non-English	English
4	Non-English	Non-English
		Note: ▪ ChatGPT can infer the language used in generate the Answer with the same langu

- Hints
  - Ask ChatGPT how to create web server code (Python Flask or node.js) to implement the UI.
    - Sample code
  - You can also refer Project: Customer Support System: Use ChatGPT to build a web-based system that can answer questions about a website
    - Python Flask
    - Node.js

C. Process for the project implementation
- Step 1: Generate a customer's comment
  - Input to ChatGPT
    - The products' detailed descriptions
  - ChaGPT's response
    - A 100 words comment about the products.
    - These words are entered into the Question part of the web interface.
  - Hint
    - This step is an automation of the task of asking ChatGPT
      -
      - The following text is the products' descriptions, please generate a
      - 100 words comment about the products.
      -
      - ===>
      -
      - [products' detailed descriptions]

    - Data collection is very critical in Data Science. This step demonstrate that we can use ChatGPT to collect data.



- Step 2: Generate email subject

        - Input to ChatGPT
            - The customer's comment created from Step 1.
        - ChaGPT's response
            - Generate the subject of an email from the customer's comment using Inferring technique.
        - Hint
            - This step is an automation of the task of asking ChatGPT
                -
                - Asuming that you provide customer support for an electronic product company.
                -
                - The following text is the customer's comment about the products, please generate a
                - subject in English of the comment. The subject will be used as the
                - subject of the email to be sent to the customer.
                -
                - ===>
                -
                - [comment]

    - Step 3: Generate the summary of the customer's comment
        - Input to ChatGPT
            - The customer's comment created from Step 1.
        - ChaGPT's response
            - Step 3.1: Generate the summary in English from the customer's comment using Summarizing technique.
        - Hint
            - This step is an automation of the task of asking ChatGPT to create the comment's summary.
                -
                - Asuming that you provide customer support for an electronic product company.
                -
                - The following text is the comment of products, please generate a
                - summary of the comment.
                -
                - Please generate an English summary of the comment
                -
                - ===>
                -
                - [comment]

- Step 4: Sentiment analysis of the customer's comment
  - Input to ChatGPT
    - The customer's comment created from Step 1.
  - ChaGPT's response
    - Sentiment analysis of the customer's comment using Inferring technique.
      - The result of the sentiment analysis shows whether the customer's comment is
        - Positive, or
        - Negative
  - Hint
    - This step is an automation of the task of asking ChatGPT to do sentiment analysis based on the comment.
      -
      - Asuming that you provide customer support for an electronic product company.
      -
      - Please do sentiment analysis based on the following comment.
      -
      - The result of the sentiment analysis shows whether the customer's comment is
      - Positive or Negative
      -
      -
      - ===>
      -
      - [comment]

- Step 5: Generate an email to be sent to the customer.
  - Input to ChatGPT
    - The customer's comment created from Step 1.
    - The Subject generetaed from Step 2 which is based on the customer's comment.
    - The summary of the customer's comment. The summary is generated from Step 3.1.
    - The result of sentiment analysis created from Step 4.
    - The customer's selected language.
  - ChaGPT's response
    - An email written in the customer's selected language. The email consists of
      1. The summary of the customer's comment. The summary is generated from Step 3.1.
      2. A response to be sent to the customer using Expanding technique.

        - The email should be put on the Answer part of the web interface for this project
    - Hint
        - This step is an automation of the task of asking ChatGPT to create an email to be sent to the customer.
            -
            - Asuming that you provide customer support for an electronic product company.
            -
            - Please create an email in [your-selected-language] to be sent to the customer based on
            -
            - 1. The customer's comment """[comment]"""
            -
            - 2. The summary of the customer's comment """[summary]"""
            -
            - 3. The result of the sentiment analysis of the customer's comment """[sentiment analysis] """
            -
            - 4. The Subject of the email """[Subject] """

    - References
        - Building an Outlook Email Reply Generator
- Step 6: send the email created in Step 5 to the customer
    - This task is optional.
    - For testing purpose, you can send email to yourself
        - Approach 1: Sending emails programmatically using Google APIs
        - Approach 2: Python code - may have password issue
    - References
        - Chapter 17 Internet Client Programming

D. Process for the project documentation

- Step 1: Adding the project to your portofolio
    1. Please use Google Slides to document the project
        - Copy from a Google Slides file and mofigy the file, but still keep the original Google Slides file.
    2. Please link your presentation on GitHub using this structure
        3.
        4. Machine Learning
        5. - ChatGPT
        6. + Customer Support System
        7. - Send an email to the customer

- Step 2: Submit
    1. The URLs of the Google Slides and GitHub web pages related to this project.
    2. A PDF file of your Google Slides
    3. Fill this form https://forms.gle/RCDrCFkvXEPCUstCA

E. References
- 2024 Spring
- 2023 Fall

Explanation:

Github Link
https://github.com/YinYinPhyo/CustomerSupportSystem-email-to-customer-?tab=readme-ov-file

Google Slide (PDF)
https://drive.google.com/file/d/15uyTaO2yJy_7Dcj51RCnCu6Ru3YPMoBO/view?usp=sharing

Create a Virtual Environment

In the terminal, navigate to the folder where you want to create your virtual environment and run the following command:

- **For Python 3**:

```
python3 -m venv myenv
```

or

```
python -m venv myenv
```

Activate the Virtual Environment

Once the virtual environment is created, you need to activate it:

- **On macOS/Linux**:

```
source myenv/bin/activate
```

Install Flask and OpenAI

Make sure Flask is installed in your virtual environment. You can do this with the following command:

```
pip install flask openai
```

Create Your Project Structure

You can organize your project files like this:

```
/your_project_directory
├── app.py              # Your Flask app
├── templates
│   └── index.html      # Your HTML file
└── static              # (Optional) For static files like CSS or images
```

Create index.html

In the templates folder, create a file named index.html. Here's a simple example:

html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Customer Support System</title>
    <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>

    <div class="container">
        <form action="/" method="post" class="form">
            <!-- Language dropdown menu -->
            <div class="language-dropdown">
                <label for="language" class="language">Choose the language of Email to be generated:</label>
                <select name="language">
                    <option value="en" {% if language == 'en' %}selected{% endif %}>English</option>
                    <option value="es" {% if language == 'es' %}selected{% endif %}>Spanish</option>
                    <option value="pt" {% if language == 'pt' %}selected{% endif %}>Portuguese</option>
```

```
            <option value="my" {% if language == 'my' %}selected{% endif %}>Burmese</option>
            <option value="zh" {% if language == 'zh' %}selected{% endif %}>Chinese</option>
            <option value="ko" {% if language == 'ko' %}selected{% endif %}>Korean</option>
        </select>
    </div>
    <div>
        <label for="translate-comment">Translate Comment in selected language:</label>
        <input type="checkbox" name="translate-comment" id="translate-comment">
        <label for="translate-email">Translate Email in selected language:</label>
        <input type="checkbox" name="translate-email" id="translate-email">
    </div>
    <input type="submit" value="Process" class="submit-button">
    <div class="content">
        {% if comment %}
        <div class="result-section">
            <div class="comment">
                <h2>Comment:</h2>
                <p>{{ comment }}</p>
            </div>
            <div class="subject">
                <h2>Subject of the email:</h2>
                <p>{{ subject }}</p>
            </div>
            <div class="summary">
                <h2>Summary of the comment:</h2>
                <p>{{ summary }}</p>
            </div>
            <div class="sentiment">
                <h2>Sentiment:</h2>
                <p>{{ sentiment }}</p>
            </div>
            <div class="email">
                <h2>Email:</h2>
                <div>
                    {{ email }}
                </div>
            </div>
```

```
            </div>
            {% endif %}
          </div>

      </form>
   </div>
</body>
</html>
```

Create app.py

In the main directory, create a file named app.py. Here's an example Flask application that serves the index.html file:

```
import os
from products import products
import utils
from openai import OpenAI
from flask import Flask, redirect, render_template, request
from dotenv import load_dotenv


app = Flask(__name__)

# Define OpenAI API_KEY
load_dotenv()

client = OpenAI()


# Define delimiter
delimiter = "####"

# Use text completion to generate the required content
def get_completion_from_messages(messages,
                                 model="gpt-3.5-turbo",
```

```
                          temperature=0,
                          max_tokens=500):
    response = client.chat.completions.create(
        model=model,
        messages=messages,
        temperature=temperature,
        max_tokens=max_tokens,
    )
    return response.choices[0].message.content


# Step 1: Generate customer comment based on the product input
def generate_customer_comment(products):

    system_message = f"""{products}"""
    user_message = f"""Generate comment in less than 100 words about the products"""

    messages =  [
    {'role':'system',
    'content': system_message},
    {'role':'user',
    'content': f"{delimiter}Assume you are a customer of the electronics company. {user_message}{delimiter}"},
    ]

    comment = get_completion_from_messages(messages)
    print('Comment:\n', comment)
    return comment


# Step 2: Generate a subject for the email from the comment
def generate_email_subject(comment):
    system_message = comment
    user_message = f"""Please generate a subject for the email from the comment using Inferring technique."""

    messages =  [
    {'role':'system',
    'content': system_message},
```

```
    {'role':'user',
    'content': f"{delimiter}Assume that you are a customer support representative of the electronics company.
{user_message}{delimiter}"},
    ]

    subject = get_completion_from_messages(messages)
    print('Subject of the email:\n', subject)
    return subject


# Step 3: Create a summary of the comment
def generate_summary(comment):
    system_message = comment
    user_message = f"""Provide a concise summary of the comment in at most 30 words."""

    messages =  [
    {'role':'system',
    'content': system_message},
    {'role':'user',
    'content': f"{delimiter}Assume that you are a customer support representative of the electronics company.
{user_message}{delimiter}"},
    ]

    summary = get_completion_from_messages(messages)
    print('Summary of the comment:\n', summary)
    return summary


# Step 4: Analyze the sentiment of the comment and tell if it is positive or negative
def analyze_sentiment(comment):
    system_message = comment
    user_message = f"""Do sentiment analysis of the comment using Inferring technique. Just mention if it is positive or
negative in one word."""

    messages =  [
    {'role':'system',
    'content': system_message},
```

```
    {'role':'user',
    'content': f"{delimiter}Assume that you are a customer support representative of the electronics company.
{user_message}{delimiter}"},
    ]

    sentiment = get_completion_from_messages(messages)
    print('Sentiment of the comment:\n', sentiment)
    return sentiment


# Translate the given content into the selected language
def get_translation(email, language):
    system_message = email
    user_message = f"""Translate the given email content into {language} using Transforming technique"""

    messages =  [
    {'role':'system',
    'content': system_message},
    {'role':'user',
    'content': f"{delimiter}{user_message}{delimiter}"},
    ]

    translate = get_completion_from_messages(messages)
    print(f"Translation of customer comment email in {language}: ")
    print(translate, "\n")
    return translate

# Step 5: Generate email based on the comment, summary, sentiment and subject generated
def generate_email(comment, subject, summary, sentiment):
    system_message = comment + subject + summary + sentiment
    user_message = f"""Create an email to be sent to the customer based on the {comment} and {sentiment}, including
{subject}, {summary} in a proper format having subject and other content."""

    messages =  [
    {'role':'system',
    'content': system_message},
    {'role':'user',
```

```
    'content': f"{delimiter}Assume that you are a customer support representative of the electronics company.
{user_message}{delimiter}"},
    ]

    email = get_completion_from_messages(messages)
    print('Email generated:\n', email)
    return email


@app.route("/", methods=("GET", "POST"))
def index():
    comment = None
    language = 'en'
    email = None
    subject = None
    summary = None
    sentiment = None

    if request.method == "POST":
        language = request.form.get("language")
        translate_comment = request.form.get("translate-comment")
        translate_email = request.form.get("translate-email")
        comment = generate_customer_comment(products)
        subject = generate_email_subject(comment)
        summary = generate_summary(comment)
        sentiment = analyze_sentiment(comment)
        email = generate_email(comment, subject, summary, sentiment)

        if translate_email:
            email = get_translation(email, language)

        if translate_comment:
            comment = get_translation(comment, language)

    return render_template('index.html', comment = comment, language = language, email = email, subject = subject,
summary = summary, sentiment = sentiment)
```

```
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=3000, debug=True)
```

Export Open AI Secrect Key

Run

```
python myScript.py
```

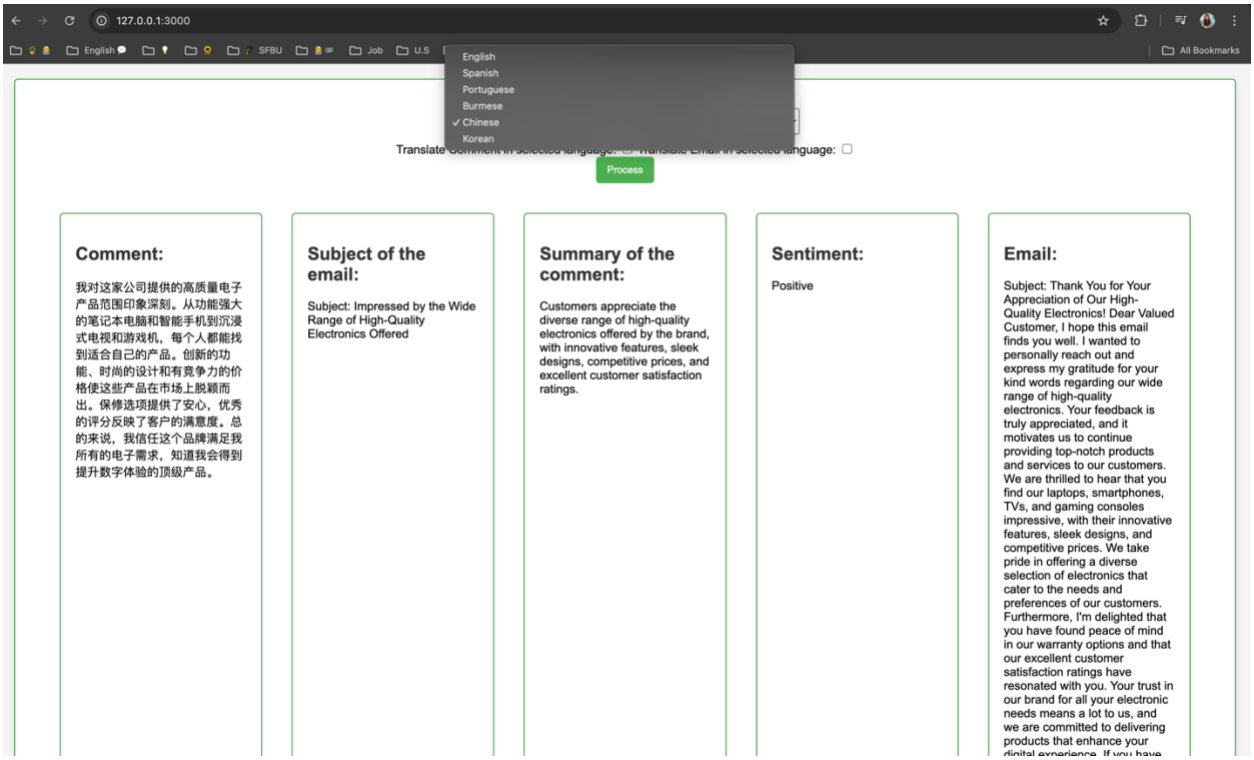
Run the Flask Server

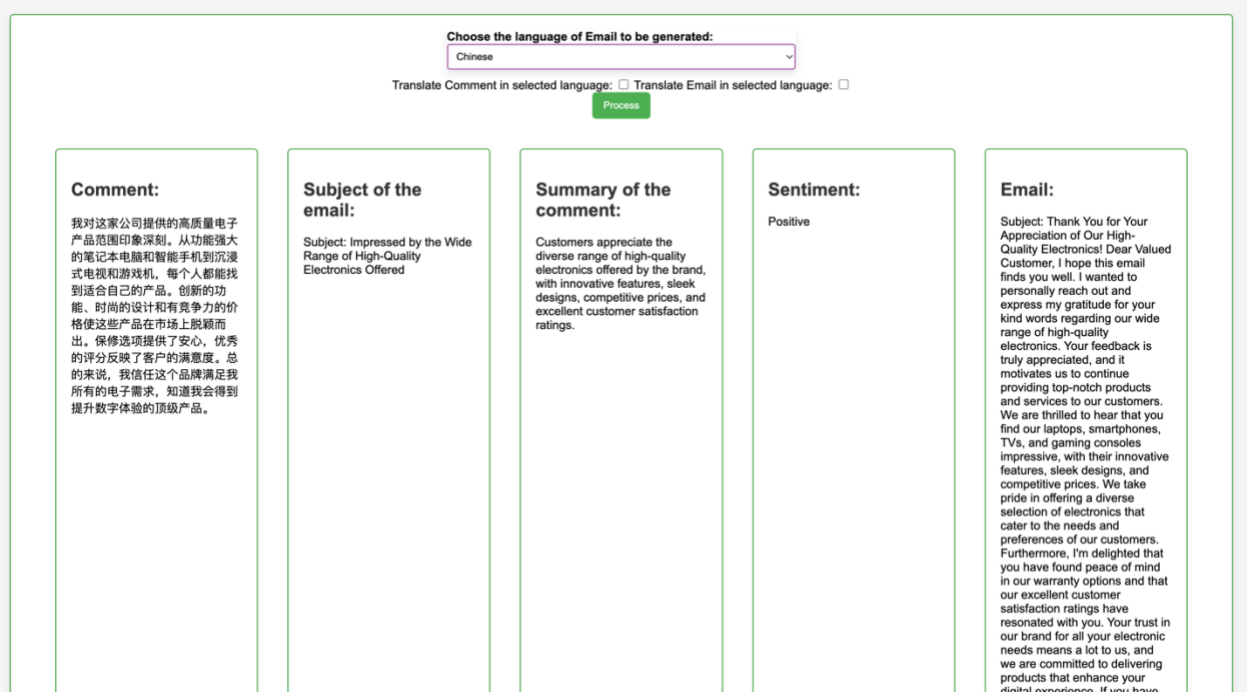
Open your terminal, navigate to your project directory, and run:

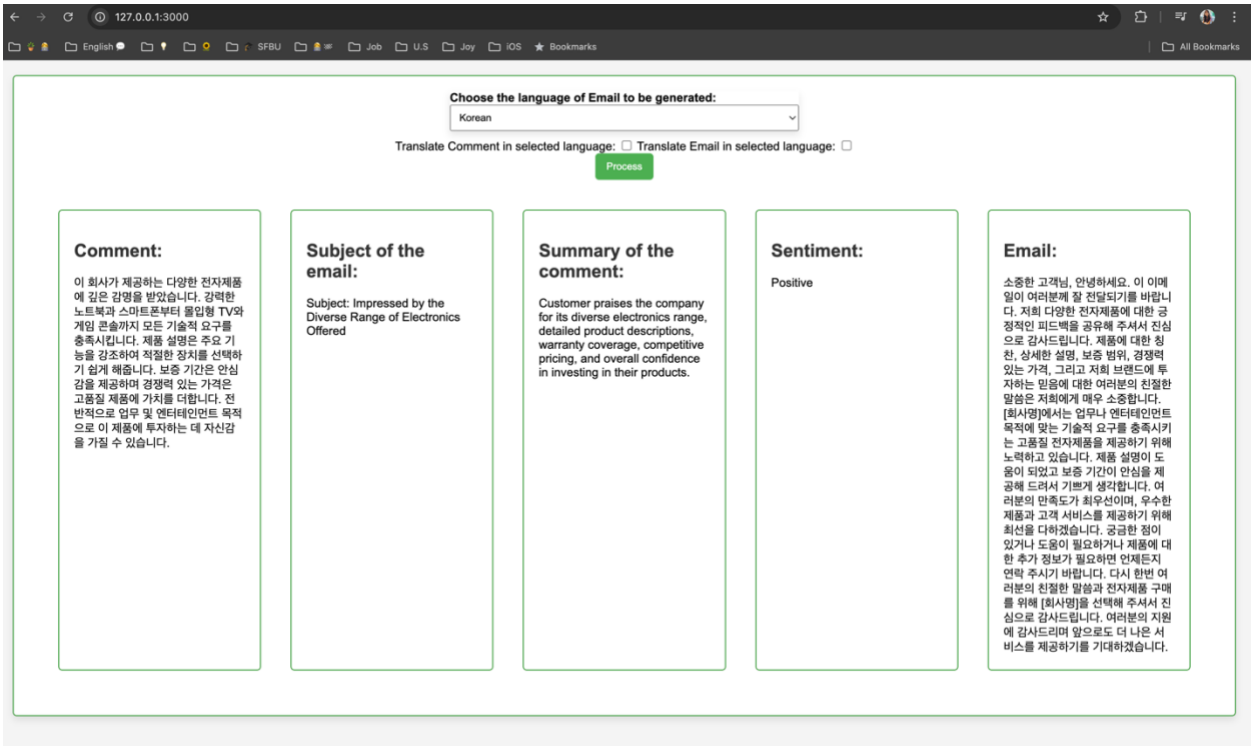
```
python3 app.py
```

Access the Application

Once the server is running, open your web browser and go to:







STEPS INVOLVED IN DEVELOPMENT:

- STEP 1: Generate customer’s comment.

    - An input of the list of products is given and expect a response of about 100 words as a customer comment.

- STEP 2: Generate email subject.
    - The comment generated is given as input and expect ChatGPT to generate appropriate subject for the email using Inferring technique.

- STEP 3: Generate summary of customer comments.
    - Based on the comment, expect ChatGPT generate a summary within 30 words.

- STEP 4: Sentiment analysis of the customer comment.
    - Take the comment as an input and expect to analyze the sentiment of the comment if it is positive or negative using Inferring technique.
    - Since it gave an output with more than 100 words, I just wanted to know if the comment is positive or negative. So, I changed the prompt accordingly.

- STEP 5: Generate email.
    - Based on all the comment, subject of email, sentiment and summary, expect to generate an email in the selected language by the user.

****